Module Title: COM6523 Software Reengineering

Assignment Report

Student Registration ID: acp21zf

Word Count: [check the assessment information on your module's Blackboard site for word count guidance]

*By attaching this cover sheet to your coursework you are confirming your agreement with the following 2 statements…*

I affirm that all sentences or passages quoted in this paper from other people’s work or my own previously submitted work (with or without trivial changes) have been placed within quotation marks, and specifically acknowledged by reference to author, work and page. I understand that plagiarism - the unacknowledged use of such passages – will be considered grounds for failure in this paper, and, if serious, in the degree examination as a whole.

I affirm that where Turnitin is used for submission I have used the FINAL Turnitin slot to check that my submission is fit for marking (including formatting and diagrams), prior to making the FINAL submission.

I understand that an anonymised version of this work may be made available to future students on Blackboard.

Table of Contents

[Module Title: COM6523 Software Reengineering 1](#_Toc100568021)

[Student Registration ID: acp21zf 1](#_Toc100568022)

[Word Count: [check the assessment information on your module's Blackboard site for word count guidance] 1](#_Toc100568023)

[1. Initial analysis 3](#_Toc100568024)

[1.1 Basic 3](#_Toc100568025)

[1.1.1 Java File Analysis 3](#_Toc100568026)

[1.1.2 LOC Analysis 3](#_Toc100568027)

[1.1.3 Commits Number Analysis 3](#_Toc100568028)

[1.2 Extra 4](#_Toc100568029)

[1.2.1 Java Mining 4](#_Toc100568030)

[1.3 Conclusion 4](#_Toc100568031)

[1.3.1 Brief Indication 4](#_Toc100568032)

[1.3.2 Stand-out Features 5](#_Toc100568033)

[1.3.3 Potential Areas of Concern 5](#_Toc100568034)

[Static analysis 5](#_Toc100568035)

[Dynamic analysis 5](#_Toc100568036)

# 1. Initial analysis

## 1.1 Basic

First, use initial-analysis.sh to call listFileSizes.sh & repMining.sh and generate csv at once. Then, analyse those csv files and save results to xlsx.

### 1.1.1 Java File Analysis

Figure 1.1 shows the LOC of .java files in main and their bar chart analysis.

Chart

Description automatically generated

Figure 1.1 LOC data and its distribution and Pareto chart. (In submission/initial-analysis/javaFileSizes.xlsx)

### 1.1.2 LOC Analysis

For the commit results, first get the count of timestamp with Pivot Table. Draw the bar chart, shown in Figure 1.2.

Graphical user interface, application

Description automatically generated

Figure 1.2 The number of commits of each file in the repo and its bar chart. (In submission/initial-analysis/joda-time.xlsx$PivotTable)

### 1.1.3 Commits Number Analysis

Matching the two columns we get Figure 1.3:

Chart, scatter chart

Description automatically generated

Figure 1.3 The correlation dot chart between LOC and counts of commits. (In submission/initial-analysis/joda-time.xlsx$Correlation)

## 1.2 Extra

## 1.2.1 Java Mining

By passing specific path or directory args into repMining.sh we get targetMining.sh.

*(In repMining.sh line 19: VERSIONS=($(git log --no-merges --pretty=format:"%h" $2)))*

It can get commits with .java file changes. The result is in submission/initial-analysis/joda-time-javaTarget.xlsx. With this we can know which classes are related during changes after some processes:

* ignore .java file with only upload.
* ignore commits related to annotation and Javadoc.
* Ignore non-functional huge commits, which include most .java files.

Finally, several classes are filtered by Excel:

Table

Description automatically generated

Figure 1.4 Java classes that are often modified and fixed in joda-time. (In submission/initial-analysis/joda-time-javaTarget.xlsx$processed3)

## 1.3 Conclusion

### 1.3.1 Brief Indication

The system has 166 java classes. Most large general-purpose classes are in org.joda.time with some abstract classes in org.joda.time.base. Specific classes are brought down the corresponding folder including chrono, convert, field, format and tz (time zone).

The number of times the Java file has been modified/committed is basically not more than 20 times.

### 1.3.2 Stand-out Features

There are many commits and thousands of files in the repo analysis. However, only 166 .java files are found in main directory.

R-squared=0.532 so the size of the .java file is in a certain proportional to the number of commits.

Most classes that are frequently updated are in org.joda.time.

### 1.3.3 Potential Areas of Concern

The difference in the number of files is a concern. The repo experienced some major changes such as movements, split and v2.0.

Classes which have more codes and frequent changes like DataTimeZone are more likely to have problems.

### 1.3.4 Source Code Analysis

The repo structure of joda-time is consistent with the indication.

The project experienced a huge change, which included splitting the project into two and overall movement, causing only 166 classes in org.joda.time.

The frequently changed classes are highly related to many other classes.

# 2. Static analysis

The whole project includes three parts, basic time format, time format conversion, and complex time format, which is based on the first two parts.

Usually the core classes (with many other classes associated) are modified less frequently to ensure the stability of the project. Classes with many modifications usually have some problems or high difficulty.

Among them, the class org.joda.time.DateTimeZone is not only very critical, but also modified many times. It is quite certain that some problems lie in this place.

# 3. Dynamic analysis